**Optimisations**

Java Virtual Machine (book)

PG.16 - JIT just in time compilation, the process of completing the compilation of a method the first time it is called (Switch statement?). Allows for faster compilation but is still slower than a language such as C, C can make better use of the registers whereas JIT relies on the slower operand stack.

PG32-35 - javap/Jasmine shows that unrolling a for loop will provide faster execution.

PG46-47 – Correct data types used, long and doubles require 64bits to store, the JVM operand stack is only 32bits and thus two positions on the stack are required which eats up more memory, Longs and doubles should only be used when absolutely necessary.

PG61-62 – locally scoped variables can be reused to prevent having to initialise new ones.

PG114 – An object is available for garbage collection when no references to it exist, an object can be set to null and then any reference to it also set to null so that the memory occupied by it can be freed.

Better, Faster, Lighter Java (book)

PG6 - Oracle warns against the use of deprecated classes.

PG9 - Cut and paste programs typically lead to bloated programs, all methods should be written from scratch.

Java – The good parts (book)

PG50 – Garbage collection

PG53 – Objects get left with references, E.g A class defines a football team, A class defines a player, The team class holds a references to the players, a player may retire and not have its reference removed from the team.

PG56 – Java does not automatically manage resources such as file handlers and sockets, when using a file handler once the reading of a file is complete the file handler should be closed. If an object is serialised sometimes people will deserialise in the constructor and then never close the file handler under the finalise method is called and the object is serialised once more, this leaves the file handler active for the duration of the program.

Hardcore Java (book)

PG38 – Common mistakes

Printing exceptions to the console can cause unreported exceptions, not all applications have a console display.

Default execution, having a default or general else clause can cause unexpected outcomes and thus should be avoided

Java revisited (Blog)

Common java multi-threading mistakes

When trying to start a new thread by using Run() instead of Start(), if Run() is used the program becomes single threaded as no new thread is created, the Start() method is what creates the new thread and then will call Run() allowing the code to run in parallel rather than sequentially.

Reset ArrayList in Java

When you wish to clear an ArrayList it is almost more efficient to use clear() over RemoveAll(), clear() gives you O(n) performance, while removeAll(Collection c) is worse, it gives O(n^2).

Online feedback directed optimisation of Java

Feedback directed splitting

Specialising sections of code within a method,

Feedback directed code in-lining

Math.max(a,b) is slower than the in-lined version (a>b)?a:b

Feedback directed code positioning

Change the order of conditional statements so that the most common occurrence is the first condition, prevents the number of instruction cache misses.

Feedback directed loop unrolling

Unrolling a for loop with a defined amount of iterations is faster, (if using a counter in a for loop this is always possible)

Writing Compliers and Interpreters, A Software Engineering Approach

Chapter 19.Additional Topics

Speed Optimisations

Constant Folding - suppose a constant is defined pi = 3.14, a function may be defined to calculate the circumference of a circle pi\*2\*radius, rather than having pi\*2 instead replace with 6.28.

Constant propagation - A variable v may be used in a function but v always contains the same value, instead of loading the value of v each time a literal constant can be used instead preventing the variable loading overhead.

Strength reduction – Replacing an operation with a faster equivalent, some operation i\*5 can be replaced with i+i+i+i+i, addition operation is faster than that of multiplication.

Loop Unrolling – Loops statements have overhead, unrolling them when they have a defined range is more efficient, don’t have to deal with incrementing counters or testing conditions.

Common Subexpression Elimination – when the same value is calculated multiple times for a function a temporary variable can be used to store that value and then used in place of the calculation, E.g x = (j+k) \* (j+k) converts to t = j=k; x = t\*t

Find a source which details

Why the JVM is bad at optimising recursion

Why using a god class when method calls to multiple classes are used infrequently

Lambda expressions over inlining